NOTES FOR 9/8/2010

Social Utility is defined as the value of the global state. In general, this is not equal to the potential function.

Examples:

$$SU(P_1, P_2 \ldots P_n) = \sum_{P_i} \sum_{e \in P_i} n_p(e)$$

$$\Phi(P_1, P_2 \ldots P_n) = \sum_{e \in E} (0 + 1 + 2 + \ldots + n_p(e))$$

Though not the same, Social Utility is related to the Potential Function. Any monadic function can be a potential function(?).

PLS: Polynomial Local Search

The material covers a lot of research papers based on congestion game

POTENTIAL GAMES AND ACHIEVING EQUILIBRIUM

Equilibrium is the state where no player wants to change their solution.

We have previously showed that the congestion game (and any other game which can be represented as a directed acyclic graph) does have a best response dynamic that will bring it to an equilibrium regardless of starting configuration.

The next question is, starting from any configuration how long will it take to converge upon an equilibrium -- assuming that the players will only make useful ($\Delta p > 0$)decisions. It is important to consider that from one configuration, multiple equilibriums could be reached, or even that the current configuration could already be an equilibrium. As a result, to achieve a meaningful solution to the proposed question, we must consider only the worst case time.

CONGESTION BASED NETWORK ROUTING GAME - WORST CASE CONVERGENCE

Given the Scheduling problem, what is the worst case time to reach equilibrium?

To answer this, we look at the bounds of the potential function.

$$\Phi(P_1, P_2 \ldots P_n) = \sum_{e \in E} (\frac{n_p(e+1) n_p(e)}{2}) \approx \sum_{e \in E} \frac{n_p(e)^2}{2} \Rightarrow [N \ldots N^2 M]$$

The maximum value this can take, if N is the number of paths and M is the number of edges, is N^2M, while the lower bound is just N. Because of the relationship between the potential function and the utility function in the scheduling game ($\Delta\Phi = \Delta p$), and the player is guaranteed to make an improvement of at least 1, so is the potential function ($\Delta\Phi \geq 1$). As a result, the worst case time to converge is the size of the bounds.

worst case steps to converge = $N^2M - N$

Cost Shared Network Routing Game - Potential Game?

The old version of the network routing game was based on a utility function based on congestion. That being, the more players using a path, the longer it takes. Players try to minimize this function. We will now consider a game which uses cost sharing utility. That being, every path has a fixed cost which is split among the players utilizing it, and the player's are trying to minimize the cost of the route. They no longer care how long the path is, just that it is the least expensive.

each hop costs $\frac{1}{k}$

k = # of users on each edge

To discover the time for convergence, we first must prove that this is in fact a potential game. To do so, we will attempt to discover a potential function using the potential function already defined for the congestion based game.

$$\Phi(P_1, P_2 \ldots P_n) = \sum_{e \in E} \left(\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n_p(e)} \right)$$

This potential function was created by observing that in the original congestion game, a similar format made it so that whenever one player removes an edge, only their cost of the edge is removed, while the inverse is true for adding an edge. In the congestion game, the cost of traversing one edge was equal to k, the number of users on an edge. In cost sharing, the cost of traversing one edge is 1/k.

The proof to prove that this is a valid potential function is also similar to the congestion game:

$$\begin{pmatrix} P_1 \\ \vdots \\ P_i \\ \vdots \\ P_n \end{pmatrix} \xrightarrow{i} \begin{pmatrix} P_1 \\ \vdots \\ P_i' \\ \vdots \\ P_n \end{pmatrix}$$

4 Cases/Events

$e \in P_i$ and $e \in P_i' \rightarrow$ no change

$e \notin P_i$ and $e \notin P_i' \rightarrow$ no change

$e \in P_i$ and : $\notin P_i' \rightarrow E-$

$e \notin P_i$ and : $\in P_i' \rightarrow E+$

When user i changes his path, there are 4 cases for every edge. An edge is either in both paths, in neither path, removed from the path, or added to the path. In the first two cases, since P_i is the only path being changed, the edge will have no effect on the potential function nor on the user's utility function. It can be seen that

$$p(P_i) = \sum_{e \in P_i} \frac{1}{n_p(e)} \qquad \Delta p(P_i, P_i') = \sum_{E+} - \sum_{E-}$$

and due to the structure of the potential function...

$$\Delta\Phi(\ldots) = \Delta u(\ldots)$$

Since the change in the potential function is always exactly equal to the change in the utlity function, this is an acceptable potential function and the cost sharing version of the network routing problem is a potential game.

Cost Shared Network Routing Game - Worst Case Convergence

Now that convergence has been proved, the question is, how fast can it converge? Its bounds are:

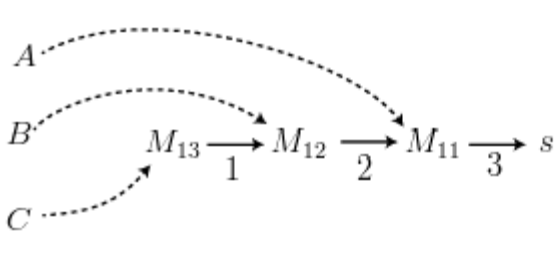
$$[\log N,\ M \log N]$$

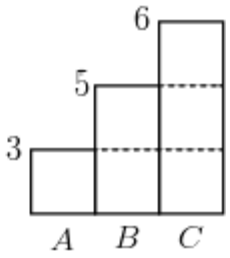
Attempting a similar method as we did for the congestion game fails because although the bounds can be found, the change in the utility function can be tiny. This is an open question. Although we have proven the existence of a potential function, and therefore that it will converge, it has not yet been proven if it will converge in polynomial time or in exponential time. An approach to solve could involve finding some repetitive patterns and then taking advantage of the fact that progress is guaranteed to be made in every step, however minute.

Reduction of Time-Shared Scheduling Game into Congestion Game

Considering 'n' jobs with multiple machines to process the jobs; we create a pipeline for each machine connected to a common sink. Each job is then connected with a 0 cost edge to the node in the machine's pipeline where its distance from the sink is equal to the time it will take for the job to complete. This technique makes for an exact mapping of the scheduling game into the congestion game.

$$\text{TimeRequired}\{A, B, C\} = \{1, 2, 3\}$$





Thus, the scheduling game can be represented as a congestion game, proving that it is a potential game and will converge.

Shortest Job First Scheduling Game - Proving Convergence

In this version of the scheduling game, each machine runs the shortest job in its queue first, and after it is done running, chooses the next shortest. To prove this is a potential game we will take a different method than previously and prove convergence rather than build a potential function.

Considering ‘shortest job first’, each of the jobs has a completion time.

$$S = (S_1 \ldots S_i \ldots S_n)$$
$$\downarrow$$
$$S' = (S_1 \ldots S'_i \ldots S_n)$$

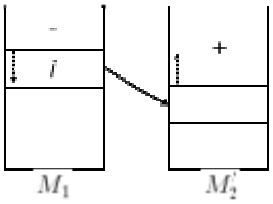
We can sort these completion times in increasing order to form a time vector.

$$I = (I_1(S) \ldots I_n(S))$$
$$\downarrow$$
$$I' = (I_1(S') \ldots I_n(S'))$$

If a job changes, the resulting time vector will be lexicographically less than the original time vector. This is because if a job decides to make a switch to be run by a different machine, it would only do so in order to achieve a better completion time.

$$(T_1 . T_2 \ldots T_r \ldots T_n)$$
$$\downarrow$$
$$(T_1 . T_2 \ldots T_s \ldots T_n)$$

So, as job I changes from time T_r to T'_r, we are assured that $T_r > T'_r$



Considering job i moving from M_1 to M_2, we can see that all the jobs that completed after job I on M1 will have their completion time decrease. The ones lower in M_1 process stack will have no change, as will the ones below the new position for job i in M_2'. Only the jobs on M2 that get pushed back by job i moving to M2 have their completion time increase, endangering our claim that the lexicographical ordering must decrease. We are only considering the lexicographical ordering of the entire time vector; the only completion time that matters is the earliest in the vector. There are two cases that must be considered.

$$(T_1 \ldots T_x, T_z \ldots T_n)$$
$$(T_1 \ldots T'_x \ldots T'_z \ldots T_n)$$

The first, if job i's original completion time (T_x) is earlier than the completion time of the job that it is pushing back T_z.. In this case, our claim holds because of our previous assertion that $T_r > T'_r$, and because only the earliest completion time that changes in the vector matters, T_z's value is irrelevant.

$$(T_1 \ldots T_z, T_x \ldots T_n)$$
$$(T_1 \ldots T'_x \ldots T'_z \ldots T_n)$$

The second case is if T_z is originally less than T_x. In order for T_z to be pushed back by T_x, the job length must of job z must be greater than the length of job i. Thus, $T'_x < T_x$ and $T'_x < T'_z$. So it can be seen that the earliest completion value in the time vector is decreasing, the lexicographical ordering is decreasing, and $S > S'$.

NOTE: This may not hold up if $T'_x = T_x$